Hydras: Complexity on general graphs and a subclass of trees

Petr Kučera*
Department of Theoretical Computer Science and Mathematical Logic
Faculty of Mathematics and Physics, Charles University in Prague
Malostranské nám. 25, 118 00 Praha 1, Czech Republic

Abstract

Hydra formulas were introduced in (Sloan, Stasi, and Turán 2012). A hydra formula is a Horn formula consisting of definite Horn clauses of size 3 specified by a set of bodies of size 2, and containing clauses formed by these bodies and all possible heads. A hydra formula can be specified by the undirected graph formed by the bodies occurring in the formula. The minimal formula size for hydras is then called the *hydra number* of the underlying graph. In this paper we aim to answer some open questions regarding complexity of determining the hydra number of a graph which were left open in (Sloan, Stasi, and Turán 2012). In particular we show that the problem of checking, whether a graph $G = (V, E)$ is single-headed, i.e. whether the hydra number of G is equal to the number of edges, is NP-complete. We also consider hydra number of trees and we describe a family of trees for which the hydra number can be determined in polynomial time.

1 Introduction

Hydra formulas were introduced in (Sloan, Stasi, and Turán 2012) as a special class of Horn formulas. A hydra formula is a definite Horn 3-CNF (i.e. conjunctive normal forms where each clause consists of exactly three literals) φ satisfying that if $(x \wedge y \rightarrow z)$ is a clause in φ then so is $(x \wedge y \rightarrow u)$ for any other variable u (except x and y). A hydra is determined by the undirected graph G formed by the bodies in φ. Given graph G we can also define its associated hydra function h_G which is defined by a hydra formula associated with G. Based on this we can define the hydra number $h(G)$ of G as the minimum number of clauses in a CNF representing h_G. Many properties of hydras were shown in (Sloan, Stasi, and Turán 2012; Stasi 2012). It is easy to see that given graph $G = (V, E)$ we have that $|E| \leq h(G) \leq 2|E|$. Graphs satisfying the lower bound are called *single-headed*.

In this paper we show that determining whether a graph is single-headed is an NP-complete problem. This answers an open question posed in (Sloan, Stasi, and Turán 2012). This result is also an interesting addition to a long line of results concerning Horn minimization problem, which is defined as follows: Given a Horn formula φ and a natural number k, determine whether there is an equivalent Horn formula ψ consisting of at most k edges. The problem of determining the hydra number of a graph is a very special case of Horn minimization problem. The Horn minimization problem for definite Horn formulas was first addressed in (Ausiello, D'Atri, and Sacca 1986) where its NP-hardness was established. Recently, it was shown in (Bhattacharya et al. 2010; Boros and Gruber 2012) that definite Horn minimization is not only hard to solve exactly but even hard to approximate even when the input is restricted to definite Horn 3-CNFs. These results imply, that definite Horn minimization is NP-hard already for 3-CNFs, a simpler proof of the same fact was recently provided in (Boros, Čepek, and Kučera 2013). However in all the definite Horn 3-CNF related results the formulas produced by the respective polynomial reductions can have prime implicates of arbitrary size. Unlike that a prime implicate of a hydra formula φ always consists of exactly three literals. NP-completeness of determining the hydra number $h(G)$ of a general graph thus implies that the following restricted version of definite 3-Horn minimization is also NP-complete: Given a definite Horn 3-CNF φ which represents a Horn function whose all prime implicates are definite Horn clauses of size 3, and a natural number k, determine whether there is an equivalent Horn CNF ψ consisting of at most k clauses.

After considering the general case we turn our attention to the complexity of determining hydra numbers of trees. Some interesting results about hydra numbers of trees were already shown in (Sloan, Stasi, and Turán 2012). It was shown in (Sloan, Stasi, and Turán 2012) that a tree $T = (V, E)$ is single-headed if and only if T is a star, and that $h(T) = |E| + 1$ if and only if T is a caterpillar. It was also shown in (Sloan, Stasi, and Turán 2012) that the hydra number of a complete binary tree $T = (V, E)$ is between $\frac{13}{12}|E|$ and $\lceil \frac{8}{7}|E| \rceil$. The complexity of determining the hydra number of a tree was left open in (Sloan, Stasi, and Turán 2012). In this paper we make first step in this direction by describing a subclass of trees such that for a tree T in this class it is possible to determine the value of $h(T)$ in polynomial time.

The paper is organized as follows. After giving necessary definitions and preliminaries in Section 2 we continue by showing the NP-completeness of determining the hydra number of general graphs in Section 3. In Section 4 we present a class of simple trees and a polynomial algorithm which determines their hydra number. In Section 5 we con-

*The author thankfully acknowledges a support by Czech Science Foundation (grant P202/10/1188).

clude the paper with some remarks and open problems. Due to the space limitations, most of the proofs are omitted.

2 Definitions

A *Boolean function* f on n propositional variables $x_1, \ldots, x_n$ is a mapping $\{0,1\}^n \to \{0,1\}$. The propositional variables $x_1, \ldots, x_n$ and their negations $\overline{x}_1, \ldots, \overline{x}_n$ are called *literals* (*positive* and *negative literals*, respectively). An elementary disjunction of literals is called a *clause*, if every propositional variable appears in it at most once. It is a well-known fact that every Boolean function f can be represented by a conjunction of clauses (see e.g. (Genesereth and Nilsson 1987)). Such an expression is called a *conjunctive normal form* (or CNF) of the Boolean function f. A clause C is an implicate of a Boolean function f if C is satisfied on all assignments which are satisfying for f, it is a prime implicate if there is no proper subclause C' of C with this property. We shall say that two CNFs are *equivalent* if they represent the same function.

A *definite Horn clause* is a clause in which exactly one literal is positive. We shall consider only the case of *definite Horn 3-clauses* which consist of three literals, one of which is positive and the other two negative, e.g. $(\overline{x} \vee \overline{y} \vee z)$, this is equivalent to implication $(x \wedge y \to z)$. The two variables appearing negatively in a definite Horn clause form its *body* and the only positive literal is called the *head* if this clause. E.g. in clause $(x \wedge y \to z)$, $\{x, y\}$ is a body of size two and z is the head. A *definite Horn (3-)CNF* is a CNF consisting of only definite Horn (3-)clauses and a *definite Horn function* is a Boolean function which can be represented by a definite Horn CNF.

In verifying that a given clause is an implicate of a given definite Horn function, a very useful and simple procedure is the following. Let φ be a definite Horn CNF of a definite Horn function h. We shall define a *forward chaining* procedure which associates to any subset Q of the propositional variables of h a set $FC_\varphi(Q)$ in the following way. The procedure takes as input the subset Q of propositional variables, initializes the set $FC_\varphi(Q) = Q$, and at each step it looks for a definite Horn clause $S \vee y$ in φ such that $S \subseteq FC_\varphi(Q)$, and $y \notin FC_\varphi(Q)$. If such a clause is found, the propositional variable y is included into $FC_\varphi(Q)$, and the search is repeated as many times as possible. The resulting set is called a *forward closure of Q with respect to φ* (we omit φ when it is clear from the context). The following lemma, proved in (Hammer and Kogan 1993), shows how the above procedure can help in determining whether a given clause is an implicate of a given CNF, or not.

Lemma 2.1 *Given a set $\mathcal{C}$ of pure Horn clauses, a subset Q of its propositional variables, and its variable $y \notin Q$, we have $y \in F_{\mathcal{C}}(Q)$ if and only if $Q \vee y$ is an implicate of the function represented by $\mathcal{C}$.*

Let us now turn our attention to hydras.

Definition 2.2 ((Sloan, Stasi, and Turán 2012)) A definite Horn 3-CNF φ is a *hydra formula*, or a hydra, if for every clause $(x \wedge y \to z)$ in φ and every variable u (other than x and y) the clause $(x \wedge y \to u)$ also belongs to φ.

For example

$$(x \wedge y \to z) \wedge (x \wedge y \to u) \wedge (x \wedge z \to y) \wedge (x \wedge z \to u)$$

is a hydra. A hydra is determined by the undirected graph G formed by the bodies in φ. Given graph G we can also define its associated hydra function h_G which is defined by a hydra formula associated with G. Based on this we can define the hydra number $h(G)$ of G as the minimum number of clauses in a CNF representing h_G. It is easy to see (Sloan, Stasi, and Turán 2012; Stasi 2012) that given graph $G = (V, E)$ we have that $|E| \leq h(G) \leq 2|E|$. Graphs satisfying the lower bound are called *single-headed*.

It is known (see (Sloan, Stasi, and Turán 2012)) that a prime implicate of a hydra formula belongs to the hydra. Let φ be a formula representing a hydra function h_G of a graph $G = (V, E)$. Let $\{a, b\} \in E$ be an edge. We say that $\{a, b\}$ is *single-headed* in φ if exactly one clause in φ has body $\{a, b\}$, otherwise $\{a, b\}$ is *multi-headed* in φ. It is now obvious that a graph G is single headed if and only if there is a representation φ of h_G in which every edge is single-headed.

Throughout the paper we shall use standard graph notation. Given graph $G = (V, E)$, the *line graph* $L(G)$ of G has vertex set $V(L(G)) = E$ and two edges $e, f \in E$ form an edge $\{e, f\} \in E(L(G))$ if they share a vertex, i.e. if $e \cap f \neq \emptyset$. A (vertex-disjoint) *path cover* of G is a set of vertex-disjoint paths such that every vertex $v \in V$ is in exactly one path. The *path cover number* of G is the smallest integer k such that G has a path cover containing k paths.

Given tree $T = (V, E)$, T^- denotes subtree of T formed by removing all leaves of T. A tree T is a *caterpillar* if T^- is a path.

3 General graphs

In this section we consider the following problem called Single-headed graph (SHG): Given an undirected graph $G = (V, E)$, is G single-headed, i.e. is $h(G) = |E|$? We show that this problem is NP-complete. It follows that a more general problem of determining the value of $h(G)$ is NP-complete as well. That both problems belong to the class NP follows from the fact that given two Horn CNFs we can check in polynomial time whether they are equivalent. This test amounts to testing whether each clause in one CNF is an implicate of the other CNF, which can be done in linear time for Horn formula (Dowling and Gallier 1984; Itai and Makowsky 1987; Minoux 1988). The rest of this section is devoted to the sketch of the proof of the following theorem:

Theorem 3.1 *Problem SHG is NP-hard.*

We shall reduce the problem of checking whether given a given cubic graph is Hamiltonian to SHG which is an NP-complete problem (Garey and Johnson 1979). Let $G = (V, E)$ be an undirected cubic graph and let $G' = (V', E')$ be constructed from G by adding a vertex in the middle of each edge. In particular, let us assume, that $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$, and let us further assume, that $e_i = \{v_{i(1)}, v_{i(2)}\}$, where $i(1)$ and $i(2)$ denote

the two endpoints of edge e_i, $1 \leq i \leq m$. Now let us define

$$\begin{aligned} V' &= V \cup \{u_1, \ldots, u_m\} \\ E' &= \Big\{\{v_{i(1)}, u_i\}, \{u_i, v_{i(2)}\} \mid 1 \leq i \leq m\Big\} \end{aligned}$$

We shall call the vertices $u_1, \ldots, u_m$ *middle vertices*. In particular given vertices $x, y \in V$, s.t. $e_i = \{x, y\} \in E$, the vertex u_i is called the *middle vertex between x and y* and we shall denote it as $u_i = mid(x, y)$.

We claim, that there is a Hamiltonian cycle in G if and only if $h(G') = |E'| = 2m$. One implication is easy and follows from the results in (Sloan, Stasi, and Turán 2012; Stasi 2012):

Lemma 3.2 *If G is Hamiltonian then G' is single-headed.*

Proof : A Hamiltonian cycle in G induces a single closed covering trail in G' and thus G' is single-headed, see (Stasi 2012; Sloan, Stasi, and Turán 2012). ■

Let us now concentrate on the opposite implication. Now let us assume that G' is single-headed. Let us observe that G' is triangle free and let us start with a technical lemma.

Lemma 3.3 *Let $R = (V_R, E_R)$ be a triangle free single-headed graph with $|V_R| \geq 4$ and let φ be a prime CNF representing the hydra function h_R associated with R. Let $\{u, v\} \in E_R$ be a single-headed body in φ and let us assume that $(u \wedge v \to w)$ is a clause within φ. Then the following propositions hold:*

(i) Either $\{u, w\} \in E_R$, or $\{v, w\} \in E_R$.

(ii) If $\{u, w\} \in E_R$ and $\{u, w\}$ is single-headed in φ, then $(u \wedge w \to v) \notin \varphi$ (the same holds symmetrically for $\{v, w\}$).

In the rest of this section let us fix a single-headed representation φ of the hydra function $h_{G'}$. In what follows we shall describe how to determine a successor of a vertex G. Given vertex $x \in V$, its successor will be denoted as $succ(x)$. This will give us a covering by pairwise vertex disjoint cycles of graph G. Later we shall show, that there is in fact only one cycle in this cover and thus it forms a Hamiltonian cycle.

Let $x \in V'$ be a vertex of G. Since G is cubic, x has exactly three neighbours a, b, c in G. Let us denote $u_a = mid(x, a)$, $u_b = mid(x, b)$, and $u_c = mid(x, c)$. Because φ represents a hydra function, vertex x is a head of some clauses of φ. By Lemma 3.3, (i) applied to a G' we have that bodies of these clauses are among $\{a, u_a\}$, $\{b, u_b\}$, and $\{c, u_c\}$. Let us assume without loss of generality, that a clause $(u_a \wedge a \to x) \in \varphi$. We can show that either b, or c is derived by forward chaining from $\{x, u_a\}$ by using only the clauses with bodies among $\{x, u_a\}, \{x, u_b\}, \{x, u_c\}$ and that we cannot derive both of b and c by using these clauses. We then define $succ(x)$ to be this derived vertex. We can show the following simple properties of $succ(x)$

Lemma 3.4 *Let $x \in V$ be arbitrary and let $y = succ(x)$, $m = mid(x, y)$. Then the following properties of $succ(x)$ are satisfied.*

(i) $succ(x)$ is uniquely defined and it does not depend on the choice of an initial clause with head x,

(ii) $x \wedge m \to y \in \varphi$, and

(iii) $\{x, y\} \in E$.

(iv) $(y \wedge m \to x)$ is not present in φ, in particular $x \neq succ(y)$

We can now show that for each $x \in V$ there is exactly one vertex $y \in V$ such that $x = succ(y)$.

Lemma 3.5 *Let $x \in V$ be arbitrary. There is exactly one vertex y for which $x = succ(y)$. Such vertex is denoted $pred(y)$ and is called the* predecessor *of y.*

Based on successor function we now define a directed graph $H = (V, A)$ on the same vertices as G. An arc (x, y) belongs to A iff $y = succ(x)$. It should be clear that the undirected version of H forms a subgraph of G, this is because the value of $succ(x)$ was selected among the neighbours of x in G. It follows from the definition of H that outdegree of every vertex in H is 1 and thus H has $n = |V|$ arcs. By Lemma 3.5 we also have that indegree of every vertex in H is 1 and thus H is a union of several pairwise vertex disjoint cycles. Moreover by Lemma 3.4, (iv) we have that each of these cycles consists of at least three vertices. It remains to show that in fact there is only one cycle in H, which then defines a Hamiltonian cycle of G. This follows from the fact that the definition of successor function is based on forward chaining and that we have to be able to derive any vertex of G' by forward chaining started from any edge of G'. In particular the following two propositions hold.

Lemma 3.6 *Let $\{x, y\}$ be an edge in G and let $m = mid(x, y)$. Let us assume that $y \wedge m \to x$ is a clause in φ. Let $z \in V$ be a vertex different from x (but possibly $z = y$). Then either $x = pred(z)$, or $pred(z)$ is added before z by forward chaining procedure starting from $\{x, m\}$.*

By iterative use of Lemma 3.6 we get the following proposition.

Lemma 3.7 *Let $\{x, y\}$ be an edge in G and let $m = mid(x, y)$. Let us assume that $y \wedge m \to x$ is a clause in φ. Let us assume that $z \in V$ is added during the forward chaining procedure started from $\{x, m\}$, then there is a path from x to z in H.*

Since φ represents a hydra function, Lemma 3.7 implies that H is strongly connected and since it is also composed of vertex disjoint cycles, it must be the case that H is actually one single Hamiltonian cycle. It follows that G is Hamiltonian and the proof of Theorem 3.1 is finished.

As a simple corollary we get that the problem of Horn minimization is NP-complete even if we restrict ourselves on formulas representing functions whose all prime implicates are cubic.

4 Trees

A useful upper bound on the value of $h(G)$ based on the path cover number of the line graph $L(G)$ was given in (Sloan, Stasi, and Turán 2012).

Theorem 4.1 ((Sloan, Stasi, and Turán 2012)) *Let $G = (V, E)$ be a connected graph and let G' be a connected spanning subgraph of G. Then the following statements are true:*

(i) If $L(G')$ is Hamiltonian then G is single-headed.

(ii) If $L(G')$ has a path cover of size k, then $h(G) \leq |E| + k$.

It was also observed in (Sloan, Stasi, and Turán 2012) that even for trees this bound is not tight. In this section we shall describe a family of simple trees for which the bound in Theorem 4.1 is tight. Because path cover of a line graph of a tree can be found in polynomial time (Raychaudhuri 1995), this will imply that we can determine the hydra number of such a simple tree in polynomial time. It was shown in (Raychaudhuri 1995) that the line graph $L(T)$ of a tree T has a path cover of size k if and only if we can find k pairwise edge disjoint caterpillars which together contain all edges of T.

Recall that we defined a caterpillar to be a tree T such that T^- is a path. Equivalently T is a caterpillar if T^- does not contain a vertex of degree 3 or more. We say that a vertex $v \in V$ which has degree at least 3 in T^- is a *critical vertex*. Critical vertices split tree T into a set of caterpillars which either connect two critical vertices, or they are adjacent to only one critical vertex. The class of simple trees for which we shall describe a polynomial algorithm for determining the hydra number is defined as follows.

Definition 4.2 A tree $T = (V, E)$ is a *simple tree* if T can be obtained from a general tree T' by subdividing each of its edges by at least one vertex such that each non-leaf edge is subdivided by at least three vertices.

If $T = (V, E)$ is a simple tree, then a subtree delimited by two critical vertices is a path which consists of at least 4 edges. Similarly a subtree delimited by one critical vertex and a leaf of T is a path which consists of at least 2 edges. The rest of this section is devoted to the sketch of the proof of the following theorem.

Theorem 4.3 *Let $T = (V, E)$ be a simple tree and let k be the size of a minimum path cover of $L(T)$, then $h(T) = |E| + k$.*

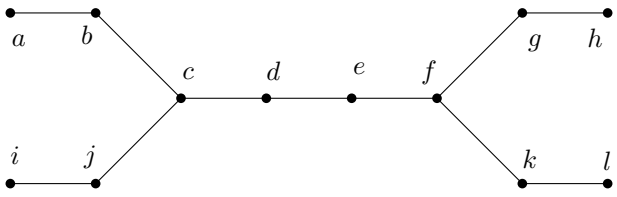

Figure 1: Example of a tree $T = (V, E)$ with path cover of $L(T)$ of size 3 and $h(T) = |E| + 2$.

The upper bound follows from Theorem 4.1, we shall thus concentrate on the opposite inequality, $h(T) \geq |E| + k$. To see that this inequality does not necessarily hold for a general tree, let us consider the tree T on Figure 1. One can check that $L(T)$ has path cover of size 3, in other words we need 3 pairwise edge disjoint caterpillars to cover all edges of T, thus $|E| + 3 = 11 + 3 = 14$. On the other hand $h(T) = 13$, consider the following formula φ representing h_T:

$$\begin{aligned}\varphi =&(a \wedge b \to c) \wedge (b \wedge c \to d) \wedge (c \wedge d \to j)\\ &\wedge (c \wedge j \to i) \wedge (i \wedge j \to k) \wedge (i \wedge j \to l)\\ &\wedge (k \wedge l \to f) \wedge (k \wedge f \to e) \wedge (f \wedge e \to g)\\ &\wedge (f \wedge g \to h) \wedge (g \wedge h \to a) \wedge (g \wedge h \to b)\\ &\wedge (d \wedge e \to c)\end{aligned}$$

Here we use the fact that after removing edge $\{d, e\}$ from T we get two disjoint caterpillars each on 5 edges, then we traverse both caterpillars for which we need only 12 clauses, then we add a clause going from $\{d, e\}$ to any neighbour, here to c. To avoid this situation we require in the definition of a simple tree that the distance between any two critical vertices is at least four edges. The other restrictions implied by Definition 4.2 are not essential and we require them only to simplify the presentation of the proof of Theorem 4.3. In other words we believe that the class of simple trees could be enlarged to the case where we only forbid paths of length 3 between two critical vertices (and we allow lengths 1 or 2). This would only make the proof more technically involved without adding any new ideas.

Let $T = (V, E)$ be a simple tree and let φ be a CNF representation of the hydra function h_T. Let $|\varphi|_c$ denote the number of clauses in φ. In the rest of this section we are going to show that T can be split into $k \leq |\varphi|_c - |E|$ edge disjoint caterpillars which together contain all edges of T. If we consider representation φ of h_T with $|\varphi|_c = h(T)$, we get that $h(T) = |\varphi|_c \geq |E| + k \geq |E| + pc(L(T))$, where $pc(L(T))$ denotes the size of path cover of $L(T)$.

In φ we shall distinguish two types of clauses, a clause $x \wedge y \to z$ is called *short* if z is a neighbour of x or y in T, it is *long* otherwise. The idea of the proof is as follows. We shall first inspect the structure of long clauses and we shall modify φ into another representation of h_T which has nicer structure of long clauses without increasing the number of clauses. Given an edge $\{x, y\}$ we shall use forward chaining restricted to using only short clauses to find a subtree of T which we can reasonably split into caterpillars. As the last step we shall use the structure of long clauses to find a suitable sequence of edges to which we should apply the restricted forward chaining to split whole T into caterpillars.

Long clauses

Let $T = (V, E)$ be a tree (the transformations described in this section work for general trees and do not require T to be simple). Let φ be a CNF representation of the hydra function h_T. Let $\{u, v\}$ be and edge and let $\Gamma^L_\varphi(u, v) = \{w \mid u \wedge v \to w$ is a long clause in $\varphi\}$ denote the *long neighbourhood of edge $\{u, v\}$ in φ*. The ultimate goal of this section is to transform φ into another CNF representation ψ of h_T with $|\psi|_c \leq |\varphi|_c$ and satisfying the following property:

Property P3: For any edge $\{u, v\}$ we have that the subgraph of T induced by $\Gamma^L_\psi(u, v)$ is either empty, or it is a matching.

We shall describe several simple and local transformations which lead to property P3. First we want to get rid of the long clauses whose heads are isolated from the others.

Lemma 4.4 *Let $T = (V, E)$ be a tree and let φ be an irredundant CNF representation of the hydra function h_T. Let $C = u \wedge v \to x$ be a long clause and let us assume that x is not necessarily used as a part of a body of a clause when deriving $FC_\varphi(\{u, v\})$, i.e. $FC_{\varphi\setminus\{C\}}(\{u, v\}) = V \setminus \{x\}$. Let y be any neighbour of x and let z be a neighbour of y other than x (note that y and z must exist due to the presence of a long clause). Let us denote*

$$\varphi' = (\varphi \setminus \{u \wedge v \to x\}) \cup \{y \wedge z \to x\}.$$

Then φ' represents h_T and $|\varphi'|_c = |\varphi|_c$.

Lemma 4.5 *Let $T = (V, E)$ be a tree and let φ be an irredundant CNF representation of the hydra function h_T. Let $C = u \wedge v \to x$ be a long clause and let us assume that x is used in a body in every forward chaining derivation of $FC_\varphi(\{u, v\}) = V$, i.e. that $FC_{\varphi\setminus\{C\}}(\{u, v\}) \subsetneq V \setminus \{x\}$. Let us assume that there is no neighbour of x which would be a head of another long clause in φ. Then there is a neighbour y of x and neighbour z of y other than x such that*

$$\varphi' = (\varphi \setminus \{u \wedge v \to x\}) \cup \{y \wedge z \to x\}$$

represents h_T and $|\varphi'|_c = |\varphi|_c$.

Note that the transformations described in lemmas 4.4 and 4.5 never introduce new long clauses. By using lemmas 4.4 and 4.5 we can now transform any (irredundant) CNF representation φ of a hydra function h_T associated with a tree T into the form satisfying the following property:

Property P1: If x is a head of a long clause, then there is a neighbour y of x which is also a head of a long clause.

Next we want the following property to be satisfied:

Property P2: If $u \wedge v \to x$ is a long clause in φ, then there is a neighbour y of x (i.e. $\{x, y\} \in E$) such that $u \wedge v \to y$ is also a long clause in φ.

Lemma 4.6 *Let T be a tree and let φ be an irredundant CNF representation of the hydra function h_T. Then φ can be in polynomial time transformed to another irredundant CNF representation ψ of the hydra function h_T satisfying property P2 and such that $|\psi|_c \leq |\varphi|_c$.*

Proof : (*Sketch*) We can assume without loss of generality, that φ does not satisfy assumptions of Lemma 4.4 and 4.5.

Let $u \wedge v \to x$ be a long clause such that there is no neighbour y of x such that $u \wedge v \to y$ would be a long clause in φ. Consider an irredundant forward chaining derivation (i.e. sequence of clauses) starting with $\{u, v\}$ using x as a part of a body. Let the first such body in this derivation be $\{x, y\}$. Thus we have that $y \in FC_{\varphi\setminus\{(u\wedge v\to x)\}}(\{u, v\})$. Let $a \wedge b \to y \in \varphi$ be a clause used to derive y. If $a \wedge b \to y$ is a short clause and say $\{b, y\} \in E$, then we can exchange $u \wedge v \to x$ for a short clause $b \wedge y \to x$. If $a \wedge b \to y$ is a long clause, then we set $\varphi' = (\varphi \setminus \{(u \wedge v \to x)\}) \cup \{(a \wedge b \to x)\}$ If $(a \wedge b \to x)$ is now a short clause and say $\{b, x\} \in E$ we further set $\varphi'' = (\varphi' \setminus \{(a \wedge b \to y)\}) \cup \{(b \wedge x \to y)\}$. ∎

If we now look at a subgraph of T induced by $\Gamma^L_\varphi(u, v)$ where φ satisfies property P2, this subgraph is a forest where no connected component is a singleton. By considering a connected component and removing leaves one by one until only one edge remains we can ensure the desired property P3.

Lemma 4.7 *Let $T = (V, E)$ be a tree and let φ be a CNF representation of the hydra function h_T. Then CNF φ can be in polynomial time transformed into an irredundant CNF ψ which also represents h_T and such that ψ satisfies property P3 and $|\psi|_c \leq |\varphi|_c$.*

Short clauses

In this subsection we shall inspect what we can get using only short clauses. The results in this section hold for general trees and not just for the simple ones.

Definition 4.8 Let $T = (V, E)$ be a tree and let φ be a CNF representation of the hydra function h_T. Let $\{u, v\} \in E$ be an edge. Let φ^S denote the CNF formed by only short clauses which appear in φ. Let $Q = FC_{\varphi^S}(\{u, v\})$ denote the set of vertices which can be reached in φ by forward chaining started in $\{u, v\}$ when we use only short clauses. Let $T^S[u, v]$ be the subgraph of T induced by the set of vertices Q.

Let us observe that $T^S[u, v]$ is always a subtree of T, in particular it is a connected subgraph. Let us introduce more notation we shall need. Let $T = (V, E)$ be a tree and let φ be a CNF representation of the hydra function h_T. Let $\{a, b\}$ be an edge, by $se(a, b)$ we shall denote the number of excess short clauses with body $\{a, b\}$, i.e. $se(a, b)$ is 0 if the number of short clauses with body $\{a, b\}$ is at most 1, or it is the number of short clauses with body $\{a, b\}$ minus 1. Let $se(T^S[u, v])$ denote the total number of excess clauses with bodies in $T^S[u, v]$, i.e. the sum of $se(a, b)$ over all edges $\{a, b\}$ in $T^S[u, v]$. Note that by definition of $T^S[u, v]$ if a short clause C has the body in $T^S[u, v]$, the head of C is also in $T^S[u, v]$.

Lemma 4.9 *Let $T = (V, E)$ be a tree and let φ be a CNF representation of the hydra function h_T. Let $\{u, v\}$ be an edge. Then we can cover the edges of $T^S[u, v]$ by at most $se(T^S[u, v]) + 1$ pairwise disjoint caterpillars.*

Proof : (*Sketch*) If $T^S[u, v]$ is a caterpillar, we are done. We shall at first assume that u is a leaf in $T^S[u, v]$. Let us fix some forward chaining derivation of $FC_{\varphi^S}(\{u, v\})$, i.e. we fix some sequence of short clauses deriving all vertices in the forward chaining closure of $\{u, v\}$. Let us now assume that a is the first critical vertex added to forward chaining closure of $\{u, v\}$ when using only short clauses. Let c be its unique neighbour on the path from u to a. (It is possible that $a = v$ and $c = u$.) Let $b_1, \ldots, b_k$ be the other neighbours of a and let us assume that T_i is a subtree of $T^S[u, v]$ rooted at a and containing only one neighbour of a, that is b_i. Let us assume that each T_i is nontrivial, i.e. it contains some edges. Trivial subtrees are omitted and then added to some caterpillar. In order to get from $\{c, a\}$ to $b_1, \ldots, b_k$ we need at least k short clauses with heads in $b_1, \ldots, b_k$. Since these are short clauses and they are used for forward chaining from $\{c, a\}$ the only possibility is that their bodies are among $\{c, a\}, \{a, b_1\}, \ldots, \{a, b_k\}$. In order to get further to $T_1, \ldots, T_k$ we need k short clauses with

bodies $\{a, b_1\}, \ldots, \{a, b_k\}$ and heads in the respective subtrees. Thus together we have $2k$ heads and $k+1$ bodies which together give us short excess at least $k-1$. By induction we get $se(T_i)+1$ caterpillars covering edges of T_i (when we start with $\{a, b_i\}$ and use only short clauses with heads in T_i). One of these caterpillars can be appended to the one we started in $\{u, v\}$. Altogether we can bound the number of caterpillars as follows:

$$\begin{aligned} \ell &\leq \sum_{i}^{k}(se(T_i)+1) = k + \sum_{i}^{k} se(T_i) \\ &= 1 + (k-1) + \sum_{i}^{k} se(T_i) = 1 + se(T^S[u,v]) \end{aligned} \tag{1}$$

In the last equality of (1) we use the fact that there are k clauses with heads $b_1, ..., b_k$, $k-1$ of them can be included in the short excess. If u is not a leaf we start the procedure on both sides of $T^S[u, v]$, i.e. we cover the u and v sides separately and then compose the results (we can save +1 in this case, as $\{u, v\}$ is necessarily a multi-headed body). ■

Splitting a tree into caterpillars

Let us fix some simple tree $T = (V, E)$ which is not a caterpillar and a formula φ representing the hydra function h_T which satisfies property P3. In this subsection we shall describe an algorithm which based on φ finds at most $|\varphi|_c - |E|$ pairwise disjoint caterpillars covering the edges of T. The idea of the algorithm is to repeatedly choose an edge $\{u, v\}$, use Lemma 4.9 to find caterpillars covering $T^S[u, v]$, take the edges of $T^S[u, v]$ out of T and then repeat with the rest of the tree. Let $e(\varphi) = |\varphi|_c - |E|$ denote the total number of excess clauses in φ. When we find a caterpillar cover of $T^S[u, v]$, we can assign an excess clause to each of $e(T^S[u, v])$ caterpillars. One caterpillar however remains without an assigned excess clause. We shall overcome this by carefully choosing the initial edge $\{u, v\}$. If there is a body $\{x, y\}$ such that $(x \wedge y \to u)$ and $(x \wedge y \to v)$ are both long clauses in φ, we have one excess clause to assign to a caterpillar containing $\{u, v\}$. The other possibility is that the caterpillar which contains $\{u, v\}$ can be attached to a caterpillar we have found earlier when considering $T^S[u', v']$ for another edge $\{u', v'\}$. We want to show that an edge satisfying one of these properties can always be found. Before that we have to consider a special case when φ does not contain any long clause.

Claim 4.10 *Let ℓ denote the number of leaves of T. If φ does not contain any long clause, then $e(\varphi) \geq \ell$). It follows that the edges of T can be covered by $\ell \leq e(\varphi)$ pairwise edge disjoint caterpillars.*

Proof : Let φ be a CNF representation of the hydra function h_T with no long clauses. Let u be any leaf of T and let v be its unique neighbour. Then v is not a critical vertex because T is simple. Thus there is another neighbour of v, let it be w. In φ we thus necessarily have clauses $(u \wedge v \to w)$, $(v \wedge w \to u)$. Since we need to get out of $\{u, v, w\}$, there must be another clause with body $\{v, w\}$, thus we can associate an excess clause $(v \wedge w \to u)$ with the leaf u. We can thus associate an excess clause with every leaf and we thus get that the value $e(\varphi)$ is at least the number of leaves in T, i.e. $e(\varphi) \geq \ell$. We can find a trivial caterpillar cover of edges of size at most the number of leaves in T. ■

In the rest of this subsection we shall assume that φ contains a long clause.

Given tree T and its subtree T' let us denote by $T \setminus T'$ the tree which is produced from T by removing edges of T' and then removing singleton connected components. Let us now consider a sequence of edges $\{u_1, v_1\}, \ldots, \{u_k, v_k\}$ and let us consider the following sequence of trees $T^0 = T$, $T^i = T^{i-1} \setminus T^S[u_i, v_i]$, $i = 1, \ldots, k$. Here $T^S[u_i, v_i]$ is taken relative to T^{i-1} (i.e. we only consider edges in T^{i-1}, it can be easily observed that even in this case each $T^S[u_i, v_i]$ is a single connected subtree).

Claim 4.11 *Let R be a connected subtree component of T^k. Then either*

- *R contains an edge $\{a, b\}$ such that both a and b are heads of long clauses with the same body, or*
- *there is an edge $\{a, b\}$ in R such that a is a leaf of R, but it is not a leaf of T, a is not a critical vertex of T, and if c is the other (than b) neighbour of a in T, then c is not a critical vertex of T either.*

Proof : Let us at first assume that R is just one edge $\{u, v\}$. If neither u, nor v is a leaf in T, then by definition of a simple tree, one of them has a neighbour which is not a critical vertex of T. Edge $\{u, v\}$ thus satisfies the second condition. If say v is a leaf in T then we set $a = u$ and $b = v$, note that a is now the only neighbour of b, because T is a simple tree. Let c be the other (than b) neighbour of a. Since edge $\{c, a\}$ is already part of $T^S[u_i, v_i]$ for some $i \leq k$, we have that there is no short clause $(c \wedge a \to b)$ in φ (otherwise $\{a, b\}$ would be included in $T^S[u_i, v_i]$ as well). Thus in order to derive b we need a long clause with head b, let it be $x \wedge y \to b$. Because φ satisfies property P3 and a is the only neighbour of b, we have that $x \wedge y \to a$ is another long clause in φ and thus the first condition is satisfied.

Now, let us assume that R is a star centered around a critical vertex b. Then b has a neighbour a in R, where a is not a critical vertex and since T is a simple tree, there is another neighbour c of a which is not a critical vertex either. The second condition is thus satisfied.

It remains to consider the case when R contains a subpath $\{x, y\}$, $\{y, z\}$, where y is not a critical vertex. Let us suppose that R does not satisfy the first condition, i.e. there is no edge $\{p, q\}$ in R such that both p and q would be heads of long clauses with the same body.

Since y is not a critical vertex and T is a simple tree, we have that x and z are the only neighbours of y. If there would be a long clause with head y, by property P3 we would have that the first condition would be satisfied. Since we assume contrary, we get that all clauses in φ having y as head are short. A short clause having head y contains either x or z in a body. If e.g. $(x \wedge w \to y)$ is a short clause in φ, then w is a neighbour of x. We can observe that edge $\{x, w\}$ must be in R, otherwise we would have that $\{x, w\}$ belongs to $T^S[u_i, v_i]$ for some $i \leq k$, which would mean that vertex

y and also edge $\{x, y\}$ would be added to $T^S[u_j, v_j]$ for some $j \leq i$. Let us now consider an irredundant forward chaining derivation of y from $\{u_1, v_1\}$ by clauses in whole φ. In order to derive y, we need to use some clauses whose bodies are in R, let $\{a, b\}$ be the first body in R derived by the above mentioned forward chaining derivation. Thus in order to derive a and b from $\{u_1, v_1\}$ we did not use any clause with body in R.

We claim that both a and b have neighbours in $T \setminus T^k = \bigcup_{i=1}^{k} T^S[u_i, v_i]$. Let us show this for a, the case of b is symmetrical. Let us distinguish the following three cases:

- If $a = u_1$ (or $a = v_1$ symmetrically), then v_1 is a neighbour of a which does not belong to R (because there would be a path using only edges in R from $a = u_1$ to v_1 which is not possible since T is a tree).
- If a was derived by a short clause $p \wedge q \rightarrow a$, then one of p and q is a neighbour of a, let us say it is q. Here edge $\{p, q\}$ does not belong to R because we do not use bodies from R to derive a. There are two cases possible.
  - If $\{p, q\}$ belongs to $T^S[u_i, v_i]$ for some $i \leq k$, then by definition of T^S edge $\{q, a\}$ belongs to $T^S[u_j, v_j]$ for $j \leq i$, in particular $\{q, a\}$ does not belong to R.
  - If on the other hand $\{p, q\}$ belongs to T^k, then it belongs to another component of T^k than R. It follows that $\{q, a\}$ cannot be an edge in R.

  From the fact that $\{q, a\}$ does not belong to R and that T is a tree we get that q does not belong to R.
- The last possible case is that a was derived using a long clause $(p \wedge q \rightarrow a)$. Because φ satisfies property P3, this implies that a has a neighbour d such that $p \wedge q \rightarrow d$ is another long clause in φ. Because R does not satisfy the first condition, we get that d does not belong to R.

Now we have that both a and b have neighbours outside of R. Since R contains at least 3 vertices we get that one of a and b has degree at least 3 in T, let us say it is b. Because T is a simple tree, it follows that b is a critical vertex in T. Thus a has degree 2 in T and it is necessarily a leaf in R. The other (than b) neighbour of a is not a critical vertex due to requirement that the distance between two critical vertices in a simple tree T is at least four edges. ■

The algorithm for determining a caterpillar cover of edges of T based on φ is described in Algorithm 4.12.

Algorithm 4.12 Caterpillar cover of a simple tree.

Input: *Simple tree $T = (V, E)$. Prime and irredundant CNF φ representing the hydra function h_T and satisfying property P3.*

Output: *Set $\mathcal{C}$ of caterpillars covering edges in T.*

1: $\mathcal{C} = \emptyset$
2: $T^0 := T$
3: $i := 0$
4: **if** φ does not contain long clauses.
5: **then**
6: Let $\mathcal{C}$ be a trivial caterpillar cover found by Claim 4.10.
7: **return** $\mathcal{C}$
8: **endif**
9: **while** T^i contains an edge
10: **do**
11: Let R be a connected component of T^i.
12: Let $\{u_{i+1}, v_{i+1}\}$ be an edge satisfying one of the two condition of Claim 4.11 (where u_{i+1} plays the role of a in the second condition).
13: Find caterpillar cover $\mathcal{C}'$ of $T^S[u_{i+1}, v_{i+1}]$ using Lemma 4.9.
14: **if** the first condition of Claim 4.11 is satisfied.
15: **then**
16: $\mathcal{C} := \mathcal{C} \cup \mathcal{C}'$
17: **else**
18: Let c denote the other (than v_{i+1}) neighbour of u_{i+1} in T.
19: Let $C_1 \in \mathcal{C}$ denote the caterpillar covering edge $\{c, u_{i+1}\}$.
20: Let $C_2 \in \mathcal{C}'$ denote the caterpillar covering edge $\{u_{i+1}, v_{i+1}\}$.
21: Let C denote the concatenation of C_1 and C_2.
22: Let $\mathcal{C} := (\mathcal{C} \setminus \{C_1\}) \cup (\mathcal{C}' \setminus \{C_2\}) \cup \{C\}$.
23: **endif**
24: Let $T^{i+1} = T^i \setminus T^S[u_{i+1}, v_{i+1}]$.
25: Let $i := i + 1$.
26: **enddo**
27: **return** $\mathcal{C}$

Let us now inspect the correctness of Algorithm 4.12.

Lemma 4.13 *Algorithm 4.12 finds caterpillar cover of all edges of T of size at most $e(\varphi)$.*

Proof : First let us show that Algorithm 4.12 finds a caterpillar cover of edges of T. This follows from the fact that when T^i is nonempty and R is a connected component of T^i, by Claim 4.11 we can find a suitable edge in step 12. At the beginning if $R = T = T^0$ we have excluded the case when there is no long clause in φ. By property P3 the whole tree $R = T$ then satisfies the first condition of Claim 4.11.

In step 24 we then remove at least the edge $\{u_{i+1}, v_{i+1}\}$ from T^i and thus we have that T^{i+1} is a proper subtree of T^i. After at most $|E|$ cycles the algorithm finishes with each edge covered by some caterpillar $\mathcal{C}$. We can also see that in step 21 we can indeed append caterpillars C_1 and C_2. In case of C_1 we have that c is not a critical vertex and thus it is a degree 2 vertex. It follows that C_1 ends with end edge $\{c, u_{i+1}\}$ and thus we can append caterpillar C_2 to C_1. In proof of Lemma **??** we have constructed C_2 so that it starts with $\{u_{i+1}, v_{i+1}\}$ and thus it can be attached to C_1 by vertex u_{i+1}.

Let us now check the size of $\mathcal{C}$. The cover $\mathcal{C}$ is modified in steps 16 and 22. In case of step 16 the number of caterpillars added to $\mathcal{C}$ is at most $es(T^S[u_{i+1}, v_{i+1}]) + 1$ by Lemma 4.9. With each but one of these caterpillars we can associate an excess clause from the short clauses considered in $es(T^S[u_{i+1}, v_{i+1}])$. The remaining caterpillar is associated with one of the two long clauses with the same body with heads u_{i+1}, v_{i+1} (it is necessarily an excess clause).

In step 22 we have that the number of caterpillars added to $\mathcal{C}$ is at most $es(T^S[u_{i+1}, v_{i+1}])$, with each of these caterpillars an excess short clause among the ones considered in $es(T^S[u_{i+1}, v_{i+1}])$ can be associated.

With each caterpillar we have uniquely associated an excess clause and thus we get that in the end $|\mathcal{C}| \leq e(\varphi)$. ■

The proposition of Theorem 4.3 now follows from Lemma 4.13.

5 Conclusion

In this paper we showed that it is NP-complete to decide whether given graph is single-headed and thus it is NP-complete to determine the value of $h(G)$ for given graph G. On the other hand we have described a class of simple trees for which the hydra number matches the upper bound based on the path cover of the line graph provided in (Sloan, Stasi, and Turán 2012). The proof of Theorem 4.3 relies mainly on the requirement that the distance of two critical vertices is at least four edges. We believe that the proof could be modified with a little (mostly technical) effort to show that the upper bound is matched by $h(T)$ even if T satisfies only this distance property.

Whether a polynomial algorithm exists for determining the hydra number of a general tree is left open for further research.

References

Ausiello, G.; D'Atri, A.; and Sacca, D. 1986. Minimal representation of directed hypergraphs. *SIAM Journal on Computing* 15(2):418–431.

Bhattacharya, A.; DasGupta, B.; Mubayi, D.; Turán, G.; Abramsky, S.; Gavoille, C.; Kirchner, C.; Meyer auf der Heide, F.; and Spirakis, P. 2010. *On Approximate Horn Formula Minimization*, volume 6198. Springer Berlin / Heidelberg. 438–450.

Boros, E., and Gruber, A. 2012. Hardness results for approximate pure horn cnf formulae minimization. In *Proceedings of International Symposium on AI and Mathematics (ISAIM)*.

Boros, E.; Čepek, O.; and Kučera, P. 2013. A decomposition method for cnf minimality proofs. *Theoretical Computer Science*.

Dowling, W., and Gallier, J. 1984. Linear time algorithms for testing the satisfiability of propositional horn formulae. *Journal of Logic Programming* 3:267 – 284.

Garey, M., and Johnson, D. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W.H. Freeman and Company.

Genesereth, M., and Nilsson, N. 1987. *Logical Foundations of Artificial Intelligence*. Los Altos, CA: Morgan Kaufmann.

Hammer, P., and Kogan, A. 1993. Optimal compression of propositional horn knowledge bases: Complexity and approximation. *Artificial Intelligence* 64:131 – 145.

Itai, A., and Makowsky, J. 1987. Unification as a complexity measure for logic programming. *Journal of Logic Programming* 4:105 – 117.

Minoux, M. 1988. Ltur: A simplified linear time unit resolution algorithm for horn formulae and computer implementation. *Information Processing Letters* 29:1 – 12.

Raychaudhuri, A. 1995. The total interval number of a tree and the hamiltonian completion number of its line graph. *Information Processing Letters* 56(6):299 – 306.

Sloan, R. H.; Stasi, D.; and Turán, G. 2012. Hydras: Directed hypergraphs and horn formulas. In Golumbic, M. C.; Stern, M.; Levy, A.; and Morgenstern, G., eds., *WG*, volume 7551 of *Lecture Notes in Computer Science*, 237–248. Springer.

Stasi, D. 2012. *Combinatorial Problems in Graph Drawing and Knowledge Representation*. Ph.D. Dissertation, University of Illinois at Chicago.